Systematische Performance-Analyse

DOAG 2012

Volker A. Brandt [1] Ulrich Gräf [2]

[1] Brandt & Brandt Computer GmbH

[2] ORACLE Deutschland B.V.& Co. KG

Vortrag am 21.11.2012

Übersicht

Übersicht

Strategie versus Taktik

Strategische Benchmarks

- Zahlen zum Veröffentlichen
- nur gute Zahlen werden veröffentlicht
- viel Finetuning

aber: **Werte in der Praxis nicht erreichbar!**

Strategie versus Taktik 2

Taktische Benchmarks

- entsprechen einem Proof-of-Concept
- aufzeigen: Applikation ist schnell genug
- Sizing festlegen
- Sicherheit beim Kunden schaffen

Taktische Performance-Analyse

Wie kann man, ohne auf die Maschine zu schauen, das Performance-Problem identifizieren?

- Disk-I/O ist das Problem **70%**
- die Applikation läuft nicht parallel **20%**

Fertig!

Nur dem Kunden muß man es noch beweisen...

Performance-Analyse: Vorgehensweise

Die klassischen Bereiche:

CPU	Sind die CPUs überlastet?
Memory	Zuwenig Memory eingebaut?
IO	Volumes/Platten zu langsam?
Netzwerk	Ist das Netzwerk überlastet?
Systemlast	Applikation überlastet das System (mit system calls, mutexes, xcalls usw.)

Übersicht

CPU-Analyse allgemein

Ist die CPU überlastet? `vmstat 5`

Spalte “idle” ist 0 (letzte Spalte rechts)
$\Longrightarrow$ 100% Auslastung – mehr geht nicht!

Bei zuviel “system time” (Spalte `sys`):
$\Longrightarrow$ Anwendung erzeugt zuviel Systemlast

CPU-Analyse: Threads

Ein Thread steht im Verdacht? `prstat -L`

Der Thread behindert die Anwendung, wenn der Wert “CPU” des Threads bei (100/*Anzahl_der_CPUs*) Prozent liegt.

Klarheit schaffen: Thread mit `pbind` an eine bestimmte CPU binden:

`pbind -b <cpu> <pid>/<thread>`

CPU-Analyse: SMP-System (mehrere CPUs oder Cores)

Eine CPU ist bei 100%? `mpstat 5` (0% wt + idl)

Prozess nutzt nur eine CPU? `prstat` (Spalte “CPU”)

Anmerkungen:

- eine CPU entspricht wieder (100/*Anzahl_der_CPUs*) Prozent
- die Spalte `wt` bei `mpstat` sagt **nichts** aus
- `prstat` zeigt die Anzahl der Threads als `NLWP`
- bei CMT-Systemen: `corestat` für Analyse notwendig

CPU-Analyse: Multiprocessing

`psrset` benutzen, um die Anwendung an eine Gruppe von CPUs zu binden: Die Anwendung bleibt dann im Prozessor-Set.

- Prozessor-Set erzeugen:

  ```
  psrset -c cpu...
  ```

- Laufende Prozesse an Prozessor-Set binden:

  ```
  psrset -b
    psrset pid...
  ```

- damit kann die CPU-Last unbeeinflußt beobachtet werden
- ein Prozessor-Set kann auch die Lösung sein!

CPU-Analyse: Behebung des Performance-Problems

Bei idle = 0% (also user + system times = 100%):

$\Longrightarrow$ mehr CPUs einbauen

$\Longrightarrow$ schnellere CPUs einbauen (wenn verfügbar, die Taktrate wächst ja immer langsamer)

$\Longrightarrow$ Anwendung anders programmieren oder benutzen:

- $\rightarrow$ mehr Threads
- $\rightarrow$ Anwendung mehrfach starten
- $\rightarrow$ Daten aufteilen in Teilmengen

$\Longrightarrow$ `psrset` / `cpuset` / `cgroup`

$\Longrightarrow$ Eigenentwicklungen: Compiler-Optionen prüfen

Übersicht

Memory-Analyse

`vmstat 5` zeigt die Spalten `swap`, `free` und `sr` (meist nicht sauber ausgerichtet). Dabei bedeuten:

swap	freier virtueller Speicher in KByte
free	freier realer Speicher in Kbyte; 1/32 muß immer frei sein (`minfree`)
sr	scan rate (dauernd > 0 bei Speichernotstand), dann mit 8k Seiten pro Sekunde Suche nach Seiten zum Freigeben; **darf nur noch bei Programmstarts > 0 sein** z.B. beim Datenbank-Start (ab Solaris 8) sonst $\Longrightarrow$ Maschine hat zuwenig Real-Speicher!

Memory-Analyse 2

Memory eines Prozesses: `pmap`

... oder mit `mdb -k`:
Macro `::memstat` verwenden

Memory-Analyse: Memory-Fresser

Buffer im Solaris können zu groß sein:

- segmap (UFS)
- ARC cache (ZFS)

Freespace wird immer benötigt; 100% des Hauptspeichers sind nicht verplanbar!
Typischerweise wird Swapping vom Anwender nicht akzeptiert (anders als früher)

$\Longrightarrow$ swapping darf maximal im Notfall auftreten!

Memory-Analyse: Virtual Memory und TLB

- virtuelle Adresse muß umgesetzt werden
- Hashtabelle im System (pagemap)

Beschleunigung durch Translation Lookaside Buffer (TLB):

- assoziativer Cache $\rightarrow$ Zugriff in einem CPU-Takt
- aber: nur 64–512 Einträge (* 8 KByte = 4 MByte!!!)
- Analyse: `cputrack`/`cpustat`(CPU-abhängige Counter)
- TLB miss (dtlb..., itlb...; siehe `cpustat -h`)
- Shared-Memory-Segmente nutzen automatisch größere Seiten (`shmat`)

Übersicht

Disk I/O: Analyse der Disk-Auslastung

`iostat -xzn 5` zeigt (nach Plattennamen sortiert):

- `read` und `write` pro Sekunde (zusammen: I/O / sec)
- `read KB` und `write KB`: Datenrate
- `asvc_t`: *average service time* in Millisekunden (ms)

Eine average service time $\leq$ 20ms ist gut, $2-5$ms ist super, $0-1$ms geht mit SSDs.

Länger ist ok, wenn der User zufrieden ist...

Disk I/O: Beispiel für `iostat -xzn 5`

```
                    extended device statistics
    r/s    w/s    kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
 1547.8    7.6 16960.3    66.4  0.0  1.0    0.0    0.7   1  78 c1t0d0
                    extended device statistics
    r/s    w/s    kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
  762.6    0.0 4612.3     0.0  0.0  0.2    0.0    0.3   0  21 c1t0d0
                    extended device statistics
    r/s    w/s    kr/s    kw/s wait actv wsvc_t asvc_t  %w  %b device
 2160.2    0.0 6312.9     0.0  0.0  0.6    0.0    0.3   1  55 c1t0d0
```

Disk I/O: Disk-Auslastung

Der Wert `%b` gibt an, zu wieviel Prozent der Zeit die Platte beschäftigt war (mindestens ein I/O aktiv).

- Das sagt gar nichts aus!
- Multiple I/Os sind bei modernen Platten möglich (tags)!

Der Wert `wait` ist $> 0 \Longrightarrow$ I/O kann nicht an Platte übertragen werden:

- entweder Limit der Platte erreicht
- oder der Wert für `max_throttle` (festgelegt in `/etc/system`) ist erreicht

Disk I/O: Weitere Analysemöglichkeiten

- Remote Mirroring über lange Strecken erhöht die Antwortzeit (zum Teil extrem)
- Aggregation des Datenverkehrs pro Controller mit `iostat -Cxn 5`
- Analyse der I/O pro Disk-Slice mit `iostat -xznp 5`

Disk I/O: Problembehebung

Bei einer Überlastung der Disk:

- $\Longrightarrow$ Daten neu verteilen
- $\Longrightarrow$ (Software-)RAID5/RAIDZ/RAIDZ2 auflösen
- $\Longrightarrow$ Plattensubsystem mit Schreibcache verwenden (wenn hohe `write`-Anzahl vorliegt)
- $\Longrightarrow$ Striping verwenden (Thin Wide Stripes!)
- $\Longrightarrow$ Storage-Subsystem mit Cache (batteriegepuffert) statt JBOD
- $\Longrightarrow$ SSD als Log oder Cache für ZFS

Disk I/O: Analyse von Volumes

- SVM-Volumes sind in `iostat` sichtbar
- Veritas-Volumes prüfen mit `vxstat`

Disk I/O: DTrace (Solaris ab 10, BSD, Mac OS X, Oracle Linux)

- read und write können der Anwendung zugeordnet werden
- Statistik über die Größe von read/write-Operationen
- welche Anwendung verursacht I/O auf
  - einer Platte?
  - allen Platten? (mit Statistik)

Disk I/O: `kstat` (ab Solaris 8)

Summarische Daten liefert `kstat`:

- Werte werden pro Platte/Partition summiert
- `iostat` nutzt dieses Interface
- welche Platten es gibt, zeigt `kstat -c disk`
- Umsetzung `sd`-Nummern in Plattennamen siehe `/etc/path_to_inst`

Übersicht

Netzwerk-I/O: Analyse

Netzwerk-Auslastung ist nicht so einfach zu sehen.
`netstat -i -I <interface> 5`
zeigt die Zahl der Pakete auf `<interface>`

`kstat -p -c net 5` zeigt alles
(ein Auswerte-Script ist nötig, um die Daten zu sortieren)

Netzwerk-I/O: Problembehebung

- $\Longrightarrow$ IPQoS (Solaris 9 und 10)
- $\Longrightarrow$ Crossbow Flows (Solaris 11)
- $\Longrightarrow$ mehr Netzwerk-Interfaces
- $\Longrightarrow$ Behebung von ungünstiger Programmierung in der Applikation
- $\Longrightarrow$ Trunking hilft nur bei 1:n-Verbindungen
- $\Longrightarrow$ 10 GBit Ethernet (demnächst 40 GBit)
- $\Longrightarrow$ umstellen von Ethernet auf Infiniband

Übersicht

Application: Systemlast

feststellbar mit `vmstat 5`:

- der Wert in Spalte sys ist zu hoch (alles ab 10%)
- schlechte Anwendungen bis zu 60%

Application: Mutex

Mutex – was ist das?

- das OS synchronisiert mit locks viele Dinge
- *spinning mutex*: Warten auf das lock, weil
  - es sich nicht lohnt, den Prozess zu suspendieren
  - man sowieso gleich drankommt (hoffentlich...)
- häufig: Werte für *spinning mutex* sind zu hoch

Application: Messen von spinning mutex

- mit `mpstat 5` in der Spalte `smtx`
  - UltraSPARC III bis 8000–12000 pro CPU
  - UltraSPARC IV bis 20000 pro CPU
  - SPARC64 50000–60000 pro CPU
  - darüber werden Systemcalls und Application langsamer
- dtrace (ab Solaris 10)

Application: spinning mutex reduzieren

Abhilfe: Anwendung ändern

$\Longrightarrow$ Anwendung synchronisiert zuviel

$\Longrightarrow$ Handshake mit zu kleinen Daten-Elementen (Byte für Byte...)

$\Longrightarrow$ zu kleine Netzwerk-Pakete

$\Longrightarrow$ Parameter möglicherweise in Konfigurations-Datei einstellbar

Application: Crosscalls

Mit einem Crosscall liest ein Prozeß auf einer CPU Daten, die von einem anderen Prozessor noch bearbeitet werden.

- tritt auf bei schlecht implementierter Anwendung
- Online-Backup, z.B. von Dataspace einer Datenbank
- Summarisch: `mpstat`, Spalte `xcal`
- Zuordnung zum Anwendungsprozeß mit DTrace

$\Longrightarrow$ Behebung: Überarbeitung des Betriebskonzepts

Application: Software-Analyse

- `truss` untersucht die Systemaufrufe von Anwendungen
- Kenntnisse der Solaris-Systemcalls sind erforderlich
- Fehlverhalten von Programmen entdeckbar (keine Garantie!)
- bessere Einstellung der Volumes oder des OS möglich

Application: Speicherzugriff

Möglicherweise springt die Anwendung „wild” durch den Hauptspeicher.

- z.B. Java...
- TLB misses und Cache-Verhalten untersuchen
- Analyse des Gesamtsystem: `cpustat`
- Analyse einer Anwendung: `cputrack`

$\Longrightarrow$ andere Pagegrößen einstellen
(der TLB der CPU wird dann besser genutzt)

Q & A

Fragen?

Volker A. Brandt, vab@bb-c.de
Ulrich Gräf, Ulrich.Graef@Oracle.com